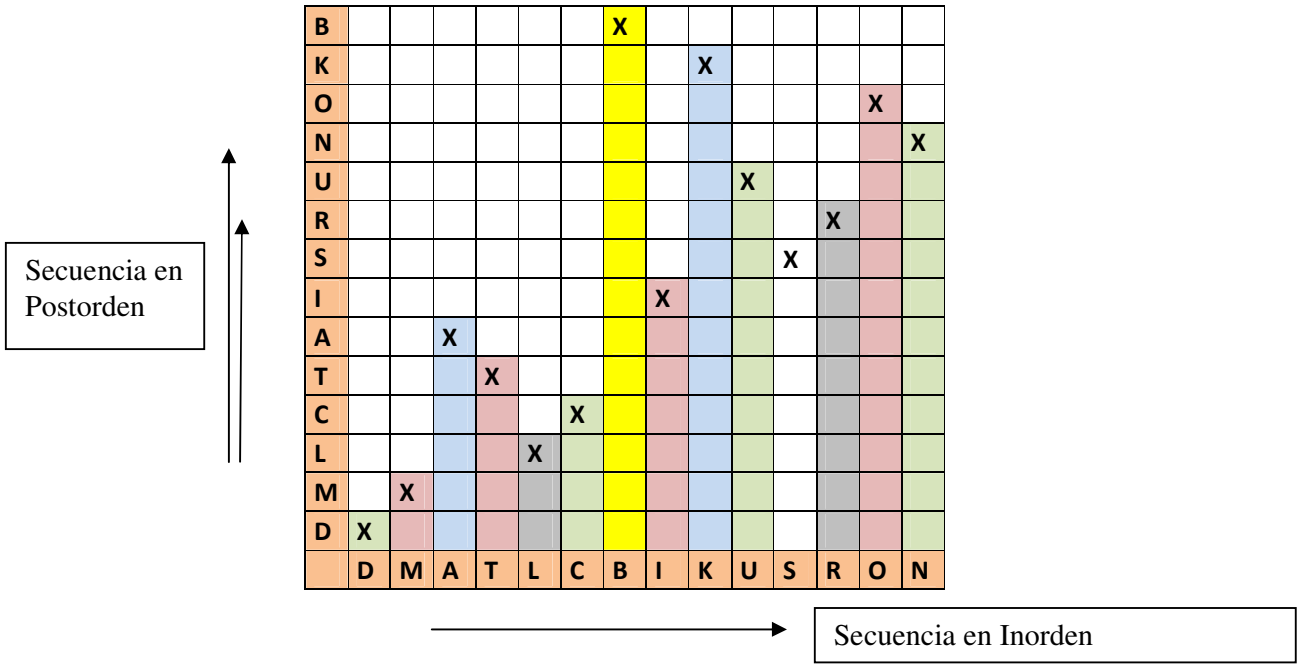
Guía resuelta de Ejercicios: Árboles.

1. El recorrido en postorden de un ABB que contiene caracteres es:
DMLCTAISRUNOKB
Y en inorden es:
DMATLCBIKUSRON
A) Dibujar el arbol binario.
B) Dar el recorrido en preorden.
C) Diseñar una funcion
void postorden(string preorden, string inorden)
Que dada la secuencia en preorden y en inorden de un árbol binario, imprima la secuencia en postorden.

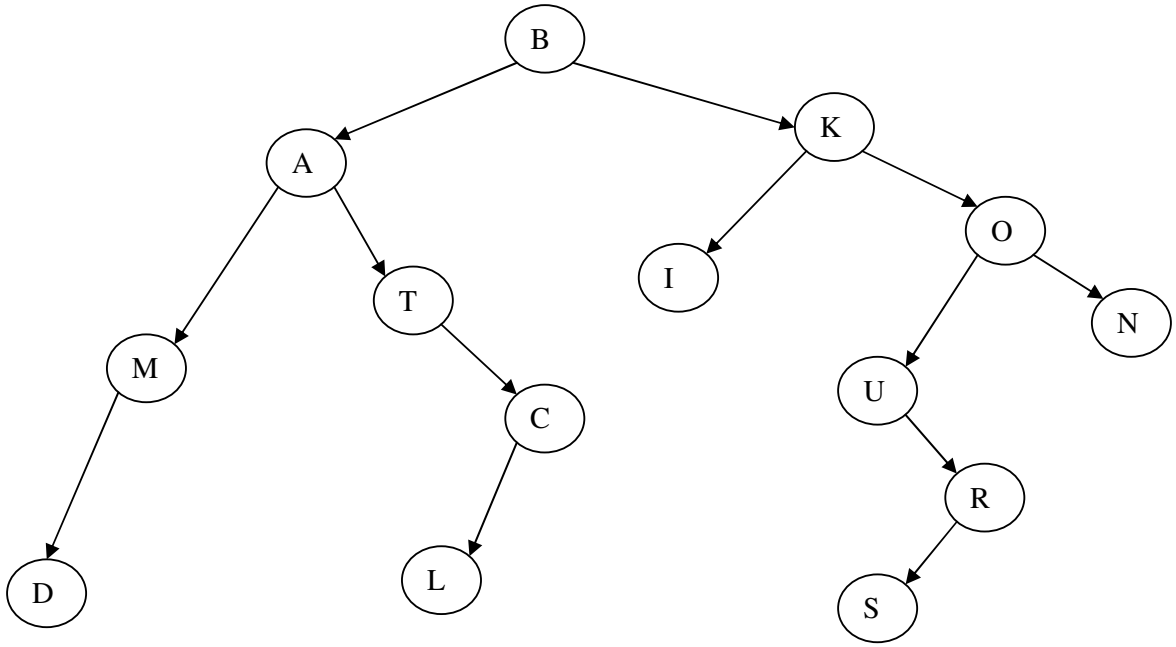
2. Se tiene un arbol AVL vacío que se le insertan, en orden, los siguientes elementos:
3, 2, 18, 5, 20, 90, 77, 40, 34, 12
A) Dibuje la disposición final del arbol AVL e indique el número de rotaciones que fueron realizadas.
B) Dibuje la disposición final de un ABB, al que se le inserta esta misma secuencia de números.
C) Justifique la eficiencia en este tipo de casos, de un AVL sobre un ABB. Use de referencia las disposiciones de las preguntas A y B.

Desarrollo:
1.A) Para poder deducir el arbol binario que tiene esos recorridos, se debe recordar que dado un inorden y un pre o postorden, existe un único arbol binario que cumple ambos recorridos.
Por lo tanto dibujaremos una tabla donde en la vertical tendremos el postorden, de abajo hacia arriba, y en la horizontal tendremos el inorden:



	D	M	A	T	L	C	B	I	K	U	S	R	O	N
B							X							
K									X					
O													X	
N														X
U										X				
R												X		
S											X			
I								X						
A			X											
T				X										
C						X								
L					X									
M		X												
D	X													

Nos damos cuenta de que existe solo una casilla por cada letra donde fila y columna coinciden.
Pero qué hacemos con esto? Construimos el árbol. La marca de más arriba será la raíz del árbol y con eso tenemos el primer nodo. Los hijos de ese nodo serán, para las sub-tablas izquierda y derecha, el nodo que esté más alto, y los hijos de esos nodos serán a su vez los nodos más altos de las sub-tablas acotadas por las raíces ya obtenidas. En el diagrama pintamos de un color los nodos de un mismo nivel. El orden de los colores es: amarillo, azul, rojo, verde, plomo, blanco.
Dibujando el árbol binario construido:



B) Luego el recorrido en preorden es intuitivo:

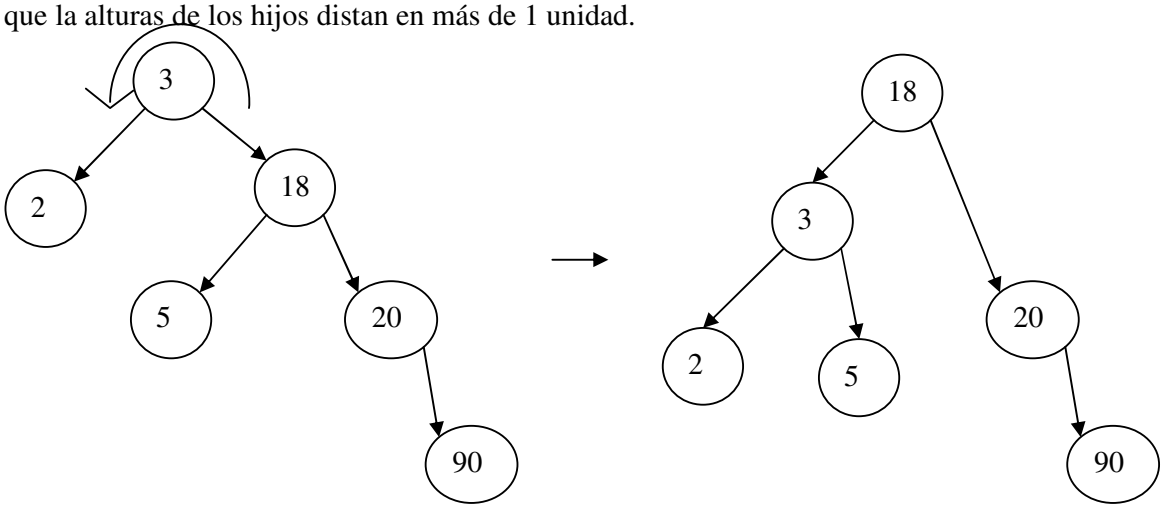
BAMDTCLKIOURSN

C) Finalmente un método que implemente esta lógica con el inorden y el preorden, sería cosa de crear la matriz, y marcar con una X los elementos donde ambos ordenes coinciden, luego armar el árbol y recorrerlo. Debemos definir un método aparte que cree un árbol a partir de la matriz. El método está anexado a esta guía en un archivo cpp. (leer el readme para usar el programa)

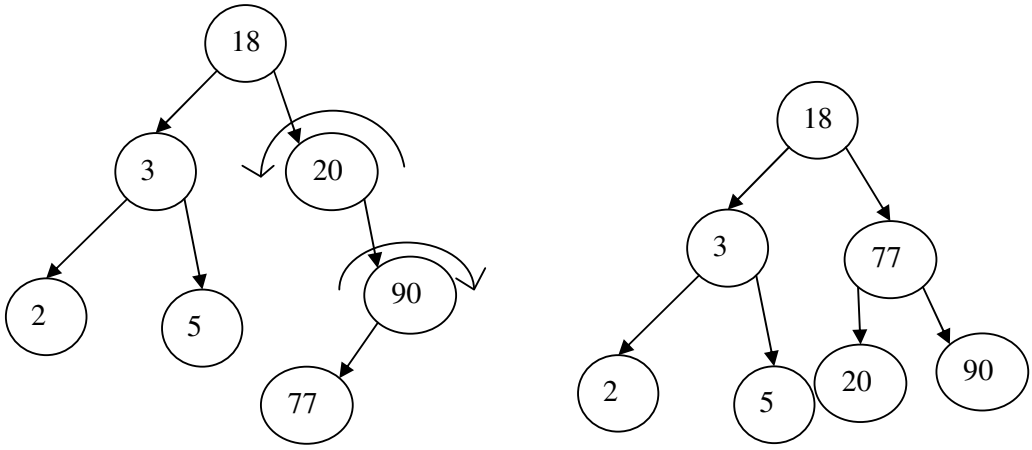
2.A) Recordando que un AVL, es un árbol de búsqueda binaria auto balanceado, iremos insertando los números y rotando.

3, 2, 18, 5, 20, 90, 77, 40, 34, 12

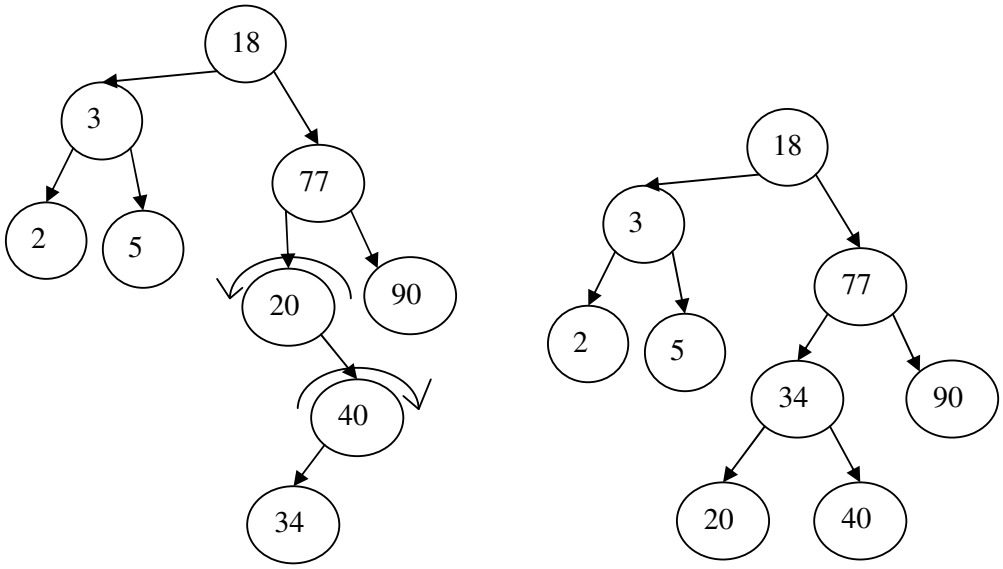
La primera rotación ocurre al insertar el 90: se rota el árbol desde la raíz, ya que ahí es donde ocurre que la alturas de los hijos distan en más de 1 unidad.



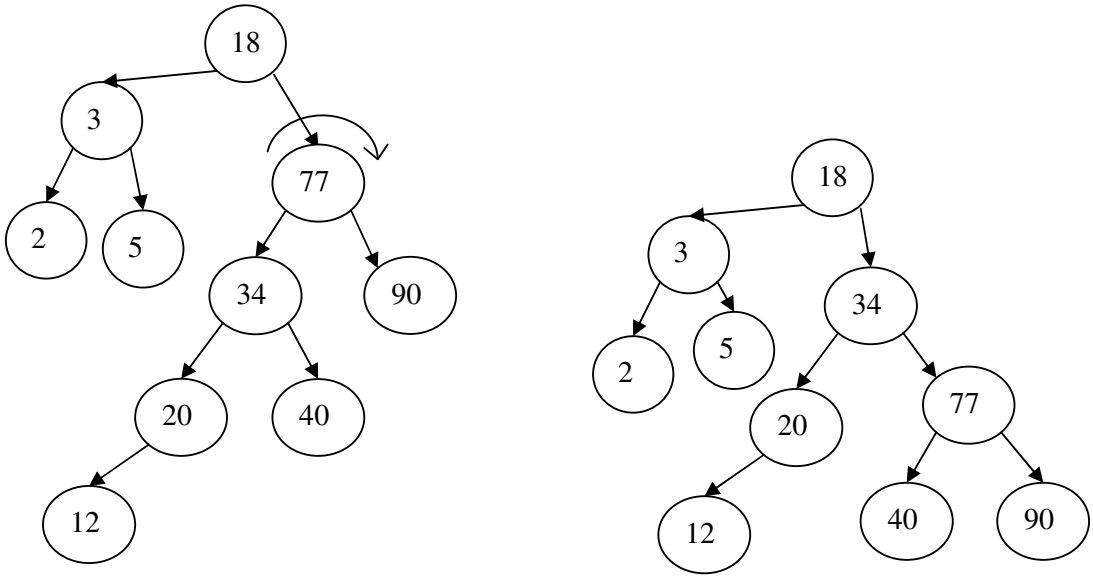
La segunda rotación ocurre inmediatamente al insertar el 77: es necesaria una rotación doble ya que fue insertado alternado al lado del padre, con respecto a la raíz del subárbol:



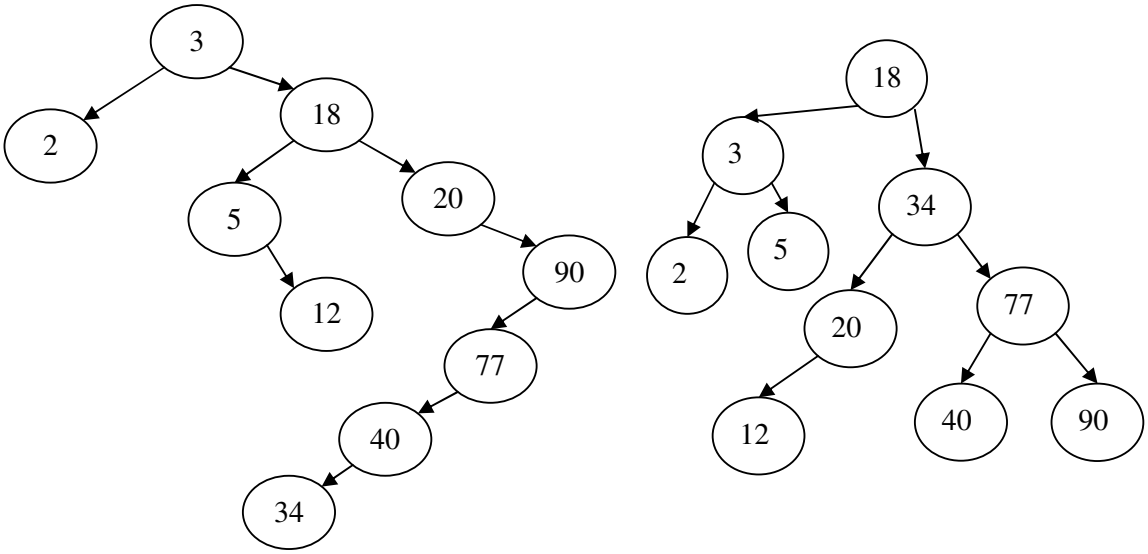
La tercera rotación ocurre al insertar el 34. Se desbalancea el subárbol de raíz 20 que se resuelve por rotación doble.



Finalmente insertar el nodo 12 produce un desequilibrio en el subárbol de raíz 77, que se resuelve por rotación simple.



B) En un ABB la disposición sería la siguiente: (para 3, 2, 18, 5, 20, 90, 77, 40, 34, 12). A la derecha el AVL del ítem anterior.



C) Primero es evidente la menor altura del AVL (3) en relación al ABB (6). Esto se traduce en un mayor tiempo de búsqueda en el ABB al tener que pasar por más nodos. El AVL, al ser auto-equilibrado, está garantizando un tiempo de búsqueda de complejidad logarítmica O(log n), deshaciéndose del caso desfavorable del ABB donde la complejidad tiende a ser lineal O(n).